;login:

The USENIX Association Newsletter

Volume 12, Number 5 **September/October 1987**

CONTENTS

The closing date for submissions for the next issue of *;login:* is October 30, 1987

USENIX **THE PROFESSIONAL AND TECHNICAL UNIX® ASSOCIATION**

NOTICE

;login: is the official newsletter of the USENIX Association, and is sent free of charge to all members of the Association.

The USENIX Association is an organization of AT&T licensees, sub-licensees, and other persons formed for the purpose of exchanging information and ideas about UNIX† and similar operating systems and the C programming language. It is a non-profit corporation incorporated under the laws of the State of Delaware. The officers of the Association are:

President	Alan G. Nemeth *usenix!agn*
Vice-President	Deborah K. Scherrer *usenix!scherrer*
Secretary	Waldo M. Wedel *usenix!wedel*
Treasurer	Stephen C. Johnson *usenix!scj*
Directors	Rob Kolstad *usenix!kolstad* Marshall Kirk McKusick *usenix!mckusick* John S. Quarterman *usenix!jsq* David A. Yost *usenix!day*
Executive Director	Peter H. Salus

The editorial staff of *;login:* is:

Editor & Publisher	Peter H. Salus *usenix!peter*
Technical Editor	Kevin Baranski-Walker *usenix!kevin*
Copy Editor	Michelle Dominijanni *{masscomp,usenix}!mmp*
Production Editor	Tom Strong *usenix!strong*

Other staff members are:

Betty J. Madden	Office Manager
Emma Reed	Membership Secretary
Jordan Hayes	Technical Consultant

USENIX Association Office
P.O. Box 2299
Berkeley, CA 94710
(415) 528-8649
{ucbvax,decvax}!usenix!office

Judith F. DesHarnais Conference Coordinator

USENIX Conference Office
P.O. Box 385
Sunset Beach, CA 90742
(213) 592-3243 or 592-1381
{ucbvax,decvax}!usenix!judy

John L. Donnelly Tutorial & Exhibit Manager

USENIX Exhibit Office
5398 Manhattan Circle
Boulder, CO 80303
(303) 499-2600
{ucbvax,decvax}!usenix!johnd

Contributions Solicited

Members of the UNIX community are encouraged to contribute articles to *;login:*. Contributions may be sent to the editors electronically at the addresses above or through the U.S. mail to the Association office. The USENIX Association reserves the right to edit submitted material.

;login: is produced on UNIX systems using *troff* and a variation of the –me macros. We appreciate receiving your contributions in *n/troff* input format, using any macro package. If you contribute hardcopy articles please send **originals** and leave left and right margins of 1″ and a top margin of 1½″ and a bottom margin of 1¼″.

Acknowledgments

The Association uses a SUN‡ 3/180S running SUN OS for support of office and membership functions, preparation of *;login:*, and other Association activities.

Connected to the SUN is a QMS Lasergrafix* 800 Printer System donated by Quality Micro Systems of Mobile, Alabama. It is used for general printing and draft production of *;login:* with *ditroff* software provided by mt Xinu.

---

†UNIX is a registered trademark of AT&T.

‡SUN is a trademark of Sun Microsystems, Inc.
*Lasergrafix is a trademark of Quality Micro Systems.

Computer Graphics Workshop

Boston Marriott Cambridge
Cambridge, MA

October 8-9, 1987

The Fourth USENIX Computer Graphics Workshop will be held at the Boston Marriott Cambridge in Cambridge, MA, October 8 and 9, 1987, with a no-host reception on the evening of October 7.

Registration will be $200 per attendee and must be paid in advance. There will be no on-site registration.

There is a special hotel rate for workshop attendees of $115 per night, single or double. Call the Marriott direct for reservations: 617-494-6600. Be sure to mention that you are a USENIX Workshop attendee. The Marriott has a strict cut-off of September 16 for its special rate. Reservations made after that date will be on a space and rate available basis.

For further program information, contact:

Tom Duff at *research!td* or Lou Katz at *ucbvax!lou.*

POSIX Portability Workshop

Berkeley Marina Marriott
Berkeley, CA

October 22-23, 1987

This USENIX workshop will bring together system and application implementors faced with the problems, "challenges," and other considerations that arise from attempting to make their products compliant with IEEE Standard 1003.

The first day of the workshop will consist of presentations of brief position papers describing experiences, dilemmas, and solutions. On the second day it is planned to form smaller focus groups to brainstorm additional solutions, dig deeper into specific areas, and attempt to forge common approaches to some of the dilemmas.

For further program information, contact:

Jim McGinness

(603) 884-5703
decvax!jmcg or jmcg@decvax.DEC.COM

C++ Workshop

Eldorado Hotel
Santa Fe, NM
November 9-10, 1987

USENIX is pleased to be hosting a workshop on the C++ language. This two-day event will bring together users of the C++ language to share their experiences. The program will be somewhat informal, comprising a mix of full presentations and shorter talks.

Bjarne Stroustrup, designer of the C++ language, has agreed to speak on topics guaranteed not to be in the C++ book, including multiple inheritance and other futures.

In conjunction with the workshop, USENIX will publish two documents:

Presentation Summaries: one or two pages from each speaker. A free copy will be available on-site to each registrant.

Full Proceedings: The proceedings will be mailed free to each registrant and will be available to the public through the USENIX office.

Vendors of products related to C++ are encouraged to send technical personnel to demonstrate their products. Call the USENIX Conference Coordinator below for more information.

The Conference Chair:

Keith Gorlen
Building 12A, Room 2017
National Institutes of Health
Bethesda, MD 20892

301-496-5363
usenix!nih-csl!keith

For registration or hotel information for any USENIX workshop, contact:

Judith F. DesHarnais
USENIX Conference Coordinator
P.O. Box 385
Sunset Beach, CA 90742

(213) 592-3243
usenix!judy

Call for Papers
Winter 1988 USENIX Conference

Dallas, Texas

February 9-12, 1988

Please consider submitting an abstract for your paper to be presented at the Winter 1988 USENIX conference. Abstracts should be around 250-750 words long and should emphasize what is new and interesting about the work. The final typeset paper should be 8-12 pages long.

The Winter conference will be four days long: two days of tutorials only and two days of papers only.

Suggested topic areas for this conference include (but are not limited to):

- Electronic Publishing
- Novel Kernels
- New Software Tools
- New Applications
- System Administration
  (including distributed systems and integrated environments)
- Security in UNIX
- Future Trends in UNIX

This conference may include a "miscellaneous" session which will include those papers which do not fit into standard tracks. Vendor presentations should contain technical information and be of interest to the general community.

Abstracts are due by **October 23, 1987**; papers absolutely must be submitted by **January 4, 1988**. Notifications of acceptance of abstracts will be sent out by November 6. Papers that do not meet the promise of their abstract will be rejected. Talks will be given on all papers published in the *Proceedings*; failure to submit a paper for an abstract will result in forfeiture of the talk.

Please contact the program chairman for additional information:

Rob Kolstad
CONVEX Computer Corporation
701 Plano Road
Richardson, TX 75081

214-952-0351 (W)
214-690-1297 (H)
214-952-0560 (FAX)

{usenix,ihnp4,uiucdcs,allegra,sun}!convex!kolstad

Please include your network address (if available) with all correspondence. It should be an ARPANET (EDUNET, COMNET), BITNET, or CSNET address or a UUCP address relative to a well-known host (e.g., *mcvax, ucbvax, decvax*, or, *ihnp4*).

Call for Papers
Summer 1988 USENIX Conference
San Francisco
June 20-24, 1988

Papers in all areas of UNIX-related research and development are solicited for formal review for the technical program of the 1988 Summer USENIX Conference. Accepted papers will be presented during the three days of technical sessions at the conference and published in the conference proceedings. The technical program is considered the leading forum for the presentation of new developments in work related to or based on the UNIX operating system.

Appropriate topics for technical presentations include, but are not limited to:

- Kernel enhancements
- UNIX on new hardware
- User interfaces
- UNIX system management
- The internationalization of UNIX
- Performance analysis and tuning
- Standardization efforts
- UNIX in new application environments
- Security
- Software management

All submissions should contain new and interesting work. Unlike previous technical programs for USENIX conferences, the San Francisco conference is requiring the submission of **full papers** rather than extended abstracts. Further, a tight review and production cycle will not allow time for rewrite and re-review. (Time is, however, scheduled for authors of accepted papers to perform minor revisions.) Acceptance or rejection of a paper will be based *solely* on the work as submitted.

To be considered for the conference, a paper should include an abstract of 100 to 300 words, a discussion of how the reported results relate to other work, illustrative figures, and citations to relevant literature. The paper should present sufficient detail of the work plus appropriate background or references to enable the reviewers to perform a fair comparison with other work submitted for the conference. Full papers should be 8-12 single spaced typeset pages, which corresponds to roughly 20 double spaced, unformatted, typed pages. Format requirements will be described separately from this call. All final papers must be submitted in a format suitable for camera-ready copy. For authors who do not have access to a suitable output device, facilities will be provided.

Four copies of each submitted paper should be received by **February 19, 1988**; this is an absolute deadline. Papers not received by this date will not be reviewed. Papers which clearly do not meet USENIX's standards for applicability, originality, completeness, or page length may be rejected without review. Acceptance notification will be by **April 4, 1988**, and final camera-ready papers will be due by **April 25, 1988**.

Send technical program submissions to:

Sam Leffler
SF-USENIX Technical Program
PIXAR
P.O. Box 13719
San Rafael, CA 94913-3719

415-499-3600
ucbvax!sfusenix

Second Distribution of Berkeley PDP-11† Software for UNIX‡

Release 2.10
(Revised April 1987)

The USENIX Association and the Computer Systems Research Group (CSRG) of the University of California, Berkeley, are pleased to announce the distribution of a new release of the "Second Berkeley Software Distribution" (2.10BSD).

This release will be handled by the USENIX association, and is available to all V7, System III, System V, and 2.9BSD licensees. The Association will continue to maintain the non-profit price of $200, as was charged by the CSRG. The release will consist of two 2400 foot, 1600 bpi tapes (approximately 80Mb) and approximately 100 pages of documentation. If you require 800 bpi tapes, please contact USENIX for more information.

If you have questions about the distribution of the release, please contact USENIX at:

2.10BSD
USENIX Association
P.O. Box 2299
Berkeley, CA 94710

+1 415 528-8649

USENIX may also be contacted by electronic mail at:

{ucbvax,decvax}!usenix!office

If you have technical questions about the release, please contact Keith Bostic at:

{ucbvax,seismo}!keith
keith@okeeffe.berkeley.edu

+1 415 642-4948

Q: What machines will 2.10BSD run on?

2.10BSD will run on:

11/24/34/44/53/60/70/73/83/84
11/23/35/40/45/50/55 with 18 or 22 bit addressing

2.10 WILL NOT run on:

T11, 11/03/04/05/10/15/20/21
11/23/35/40/45/50/55 with 16 bit addressing

Q: What's new in this release?

Lots of stuff. This release is 4.3BSD. We don't expect to distribute manuals this time, we expect people to simply use the 4.3BSD ones. A list of some of the larger things that have been added:

22-bit Qbus support
4.3BSD networking, (TCP/IP, SLIP)
4.3BSD serial line drivers
4.3BSD C library

† DEC, PDP, and VAX are trademarks of Digital Equipment Corporation.

‡ UNIX is a trademark of Bell Laboratories.

most of the 4.3BSD application programs
complete set of 4.3BSD system calls
MSCP device driver for (RQDX? UDA50, KLESI)
RAM disk
inode, core, and swap caching
conversion of the entire system to a 4.3BSD structure

Q: Why get this release?

You want to get this release for one of two reasons. Either you have a number of 4.3BSD programs or machines in your environment and you'd like consistency across the environment, or you want a faster, cleaner version of 2.9BSD, with or without networking.

This release is, without question, considerably faster than any other PDP-11 system out there. There have been several major changes to the 2.10BSD kernel to speed it up.

- The kernel *namei* routine has been modified to read the entire path name in at once rather than as a single character at a time, as well as maintaining a cache of its position in the current directory.
- The *exec* routine now copies its arguments a string at a time, rather than a character at a time.
- All inodes are placed in an LRU cache, eliminating going to disk for often used inodes; kernel inodes also contain more of the disk inode information to eliminate require disk access for *stat*(2) calls.
- Both core and swap are LRU cached; the former is particularly interesting on PDP-11's with large amounts (for PDP's, anyway) of memory. Our experience with an 11/44 with 4M of memory, in a student environment, is that it never swaps, and only rarely do programs leave core.

  This change is largely responsible for My Favorite Timing: Ultrix 11, V3.0, on my 11/73, with a single RD52, takes 1.1 system seconds to run *vi*. 2.9BSD takes approximately .9 system seconds, a difference probably attributable to the fact that 2.9BSD has *vfork*. Once 2.10BSD has the *vi* image in its core cache, it executes *vi* in .2 system seconds.
- Finally, many other speedups, such as rewriting several of C library routines in assembler, replacing the kernel clist routines with the faster 4.3BSD ones, caching and hashing process id's, and splitting the process list into multiple lists have been added.

Q: How good is the networking?

The networking is 4.3BSD's. It runs, it runs correctly. It eats space like there's some kind of reward. We are considering fixing the latter by moving the networking into supervisor space.

Q: Will this release be supported?

This release is not supported, nor should it be considered an official Berkeley release. It was called 2.10BSD because 2.9BSD has clearly become overworked and System V was already taken.

The "bugs" address supplied with this release (as well as with the 4BSD releases) will work for some unknown period of time; make sure that the "Index:" line of the bug report indicates that the release is "2BSD." See the *sendbug*(8) program for more details. All fixes that we make, or that are sent to us, will be posted on USENET, in the news group *comp.bugs.2bsd*. USENIX is aware of this problem and is willing to make hard-copy bug reports available to those of you not connected to the net.

To summarize, all that I can say is that any *major* problems will be fixed, i.e. if you've got a program that's crashing the kernel, we'll be inclined to fix it. If *ls* is misformatting its output, you're probably on your own.

Q: Is this the last release?

Yes, at least by us; quite frankly, we'd rather sacrifice our chance at heaven than look at a 16-bit machine again.

Q: Who did all this wonderful, exciting, neat stuff?

Mostly Casey Leedom, of California State University, Stanislaus, and Keith Bostic, of the CSRG. From the "Changes to the Kernel in 2.10BSD" paper:

> The authors gratefully acknowledge the contributions of many other people to the work described here. Major contributors include Gregory Travis of the Institute for Social Research, and Steven Uitti of Purdue University. Jeff Johnson, also of the Institute for Social Research, was largely responsible for the port of 4.3BSD's networking to 2.10BSD. Cyrus Rahman of Duke University should hold some kind of record for being able to get the entire kernel rewritten with a single 10-line bug report. Much credit should also go to the authors of 4.2BSD and 4.3BSD from which we stole everything that wasn't nailed down and several things that were. (Just *diff* this document against *Changes to the Kernel in 4.2BSD* if you don't believe that!) We are also grateful for the invaluable guidance provided by Michael Karels, of the Computer Science Research Group, at Berkeley – although we felt that his suggestion that we "just buy a VAX," while perhaps more practical, was not entirely within the spirit of the project.

The tape that USENIX will be distributing for the first few weeks will **only** support machines with split I/D and floating point hardware. This is not because any work remains to be done, but because we just haven't had the time to build and test a system.

Sites wishing to run 2.10BSD should also be aware that the networking is only lightly tested, and that certain hardware has yet to be ported. Contact Keith Bostic at the above address for current information as to the status of the networking. As of August 6, 1987, the complete 4.3BSD networking is in place and running, albeit with minor problems. The holdup is that only the Interlan Ethernet driver has been ported, as well as some major space constraints. Note, if we decide to go to a supervisor space networking, 2.10 networking will only run on:

11/44/53/70/73/83/84
11/45/50/55 with 18 bit addressing

Keith Bostic
Casey Leedom

Manuals Now Available to All Members!

After a long period of negotiation with both AT&T and the Regents of the University of California, the USENIX Association is now fully licensed for both System V and 4.3BSD. This means that the Association can now offer *all* members of the Association the opportunity to purchase 4.3BSD manuals.† This article provides information on the contents, cost, and ordering of the manuals.

The 4.3BSD manual sets are significantly different from the 4.2BSD edition. Changes include many additional documents, better quality of reproductions, as well as a new and extensive index. All manuals are printed in a photo-reduced 6″×9″ format with individually colored and labeled plastic "GBC" bindings. All documents and manual pages have been freshly typeset and all manuals have "bleed tabs" and page headers and numbers to aid in the location of individual documents and manual sections.

A new Master Index has been created. It contains cross-references to all documents and manual pages contained within the other six volumes. The index was prepared with the aid of an "intelligent" automated indexing program from Thinking Machines Corp. along with considerable human intervention from Mark Seiden. Key words, phrases and concepts are referenced by abbreviated document name and page number.

While two of the manual sets contain three separate volumes, you may only order complete sets.

The costs shown below do not include applicable taxes or handling and shipping from the publisher in New Jersey, which will depend on the quantity ordered and the distance shipped. Those charges will be billed by the publisher (Howard Press).

Manuals are available now. To order, return a completed "4.3BSD Manual Reproduction Authorization and Order Form" to the USENIX office along with a check or purchase order for the cost of the manuals. You **must** be a USENIX Association member. Checks and purchase orders should be made out to "Howard Press." The manuals will be shipped to you directly by the publisher.

Manual	Cost*
User's Manual Set (3 volumes) User's Reference Manual User's Supplementary Documents Master Index	$25.00/set
Programmer's Manual Set (3 volumes) Programmer's Reference Manual Programmer's Supplementary Documents, Volume 1 Programmer's Supplementary Documents, Volume 2	$25.00/set
System Manager's Manual (1 volume)	$10.00

* Not including postage and handling or applicable taxes.

4.2BSD Manuals are No Longer Available

† Tom Ferrin of the University of California at San Francisco, a former member of the Board of Directors of the USENIX Association, has overseen the production of the 4.2 and 4.3BSD manuals.

4.3BSD Manual Reproduction Authorization and Order Form

This page may be duplicated for use as an order form

USENIX Member No.: _____

Purchase Order No.: _____

Date: _____

As a USENIX Association Member in good standing, and pursuant to the copyright notice as found on the rear of the cover page of the UNIX®/32V Programmer's Manual stating that

> "Holders of a UNIX®/32V software license are permitted to copy this document, or any portion of it, as necessary for licensed use of the software, provided this copyright notice and statement of permission are included,"

I hereby appoint the USENIX Association as my agent, to act on my behalf to duplicate and provide me with such copies of the Berkeley 4.3BSD Manuals as I may request.

Signed: _____

Institution (if Institutional Member): _____

Ship to:

Name: _____

_____

_____

_____

_____

Phone: _____

Billing address, if different:

Name: _____

_____

_____

_____

_____

Phone: _____

The prices below **do not** include shipping and handling charges or state or local taxes. All payments must be in US dollars drawn on a US bank.

4.3BSD User's Manual Set (3 vols.)	_____	at $25.00 each	= $_____
4.3BSD Programmer's Manual Set (3 vols.)	_____	at $25.00 each	= $_____
4.3BSD System Manager's Manual (1 vol.)	_____	at $10.00 each	= $_____
Total	_____		$_____

[ ] Purchase order enclosed; invoice required.
(Purchase orders **must** be enclosed with this order form.)

[ ] Check enclosed for the manuals: $_____
(Howard Press will send an invoice for the shipping and handling charges and applicable taxes.)

Make your check or purchase order out to "Howard Press" and mail it with this order form to:

Howard Press
c/o USENIX Association
P.O. Box 2299
Berkeley, CA 94710

for office use: m.v.: _____ check no.: _____ amt. rec'd: _____

RT PC Distributed Services: File System

Charles H. Sauer
Don W. Johnson
Larry K. Loucks
Amal A. Shaheen-Gouda
Todd A. Smith

IBM Industry System Products
Austin, Texas 78758

Introduction

RT PC Distributed Services (RT/DS) provides distributed operating system capabilities for the AIX[1] operating system. These include distributed files with local/remote transparency, a form of "single system image" and distributed interprocess communication. The distributed file design supports "traditional" AIX and UNIX[2] file system semantics. This allows applications, including data management/data base applications, to be used in the distributed environment without modification to existing *object* code. The design incorporates IBM architectures such as SNA and some of the architectures of Sun Microsystems'[3] NFS. This paper focuses on key characteristics and decisions in the design of the Distributed Services file system.

There have been numerous research efforts and a number of products with goals and characteristics similar to the RT/DS file system. Perhaps the best known of these are LOCUS [Popek *et al* 1981, Popek and Walker 1985], NFS [Sandberg *et al* 1985, Sun 1986] and RFS [Rifkin *et al* 1986]. We studied each of these, and many other, previous designs. As we describe the RT/DS file system, we will contrast our design with some of its predecessors.

Administrative Environments

Assumptions about administrative environments are fundamental to design of distributed systems, yet administrative concerns are often omitted from primary consideration. Two primitive elements can be identified in the environments we anticipate: multi-machine clusters and independently administered machines.

Multi-Machine Clusters

All of the machines[4] in a multi-machine cluster are administered uniformly, let us assume by the same person. The machines in the cluster are likely owned by the same department, and members of the department can use the machines equivalently, i.e., the multiple machines present a "single system image." Regardless of which machine in the cluster is used, login ids and passwords are the same, the same programs/data files/directories are accessible and authorization characteristics are the same. The large boxes marked D46, D29, ... in Figure 1 are meant to suggest multi-machine clusters, with each small box representing a machine. Of course, the machines may be dispersed geographically within the limitations of the networks used – they are shown together in a room for convenience in drawing the figure.

[1] AIX is a trademark of International Business Machines Corporation.

[2] Developed and licensed by AT&T. UNIX is a registered trademark in the U.S.A. and other countries.

[3] Sun Microsystems is a trademark of Sun Microsystems, Inc.

[4] We use "machine" to indicate a generic computer – a personal computer, workstation or larger computer.

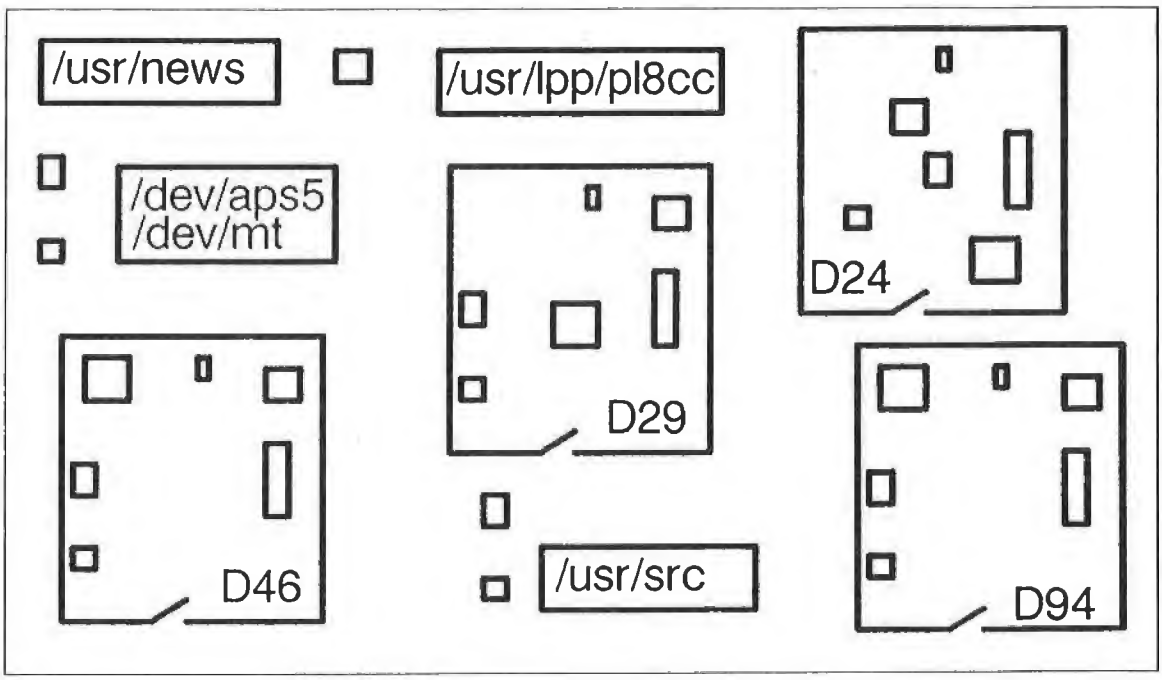

Figure 1. Multi-machine clusters and separately administered machines.

Independently Administered Machines

The other primitive element is the independently administered machine. These machines fall into two subcategories: servers and private machines. Servers in this sense are not functionally different from other servers (e.g., file or device servers) which may be found within multi-machine clusters, but they are administered independently from other machines/clusters. The boxes with path names in Figure 1 are intended to suggest file/device servers administered independently. Other machines may be independently administered because the owners are unwilling to allow others to assume administrative responsibility. Both of these subcategories can be considered degenerate cases of the multi-machine cluster, but it is convenient to discuss them separately.

Networks of Multi-Machine Clusters/Independently Administered Machines

As organizations evolve toward connecting all machines with multimegabit per second networks, administrative configurations such as the one depicted in Figure 1 will inevitably occur. It will be required that all of the machines be able to communicate with one another, and a high degree of network transparency will be required. But administrative clustering of machines according to subgroups of the organization will be natural, and cooperation/transparency within these clusters will usually be a primary issue. Authorization characteristics will vary across the clusters/independent machines. Organizations will change, and correspondingly, machines will be added to/deleted from clusters, and clusters/machines will be added to/deleted from networks. Distributed system designs must be prepared to cope with these configurations and changes in configuration.

Distributed Services Design Goals

The primary design goals in our design of Distributed Services were

Local/Remote Transparency in the services distributed. From both the users' perspective and the application programmer's perspective, local and remote access appear the same.

Adherence to AIX Semantics and UNIX Operating System Semantics. This is corollary to local/remote transparency: the distribution of services cannot change the semantics of the services. Existing object code should run without modification, including data base management and other code which is sensitive to file system semantics.

Remote Performance = Local Performance. This is also corollary to transparency: if remote access is noticeably more expensive, then transparency is lost. Note that caching effects can make some distributed operations *faster* than a comparable single machine operation.

Network Media Transparency. The system should be able to run on different local and wide area networks.

Mixed Administrative Environments Supported. This was discussed in the previous section. Additionally, services must be designed to make the administrator's job reasonable.

Security and Authorization Comparable to a Single Multiuser Machine.

RT/DS File System

Remote Mounts

Distributed Services uses "remote mounts" to achieve local/remote transparency. A remote mount is much like a conventional mount in the UNIX operating system, but the mounted filesystem is on a different machine than the mounted on directory. Once the remote mount is established, local and remote files appear in the same directory hierarchy, and, with minor exceptions, file system calls have the same effect regardless of whether files(directories) are local or remote[5]. Mounts, both conventional and remote, are typically made as part of system startup, and thus are established before users login. Additional remote mounts can be established during normal system operation, if desired.

Conventional mounts require that an entire file system be mounted. Distributed Services remote mounts allow mounts of subdirectories and individual files of a remote filesystem over a local directory or file, respectively. File granularity mounts are useful in configuring a single system image. For example, a shared copy of `/etc/passwd` may be mounted over a local `/etc/passwd` without hiding other, machine specific, files in the `/etc` directory. Directory granularity and file granularity mounts are now also allowed with AIX local mounts.

Distributed Services does not require a file system server to export/advertise a file system before it can be mounted. If a machine can name a directory/file to be mounted (naming it by node and path within that node), then the machine can mount the directory/file if it has the proper permissions. The essential permission constraints are

1. Superuser (root) can issue any mount.
2. System group[6] can issue local device mounts defined in the profile `/etc/filesystems`.

[5] The traditional prohibition of links across devices applies to remote mounts. In addition, Distributed Services does not support direct access to remote special files (devices) and the remote mapping of data files using the AIX extensions to the `shmat()` system call. Note that program licenses may not allow execution of a remotely stored copy of a program.

[6] In AIX, we have given the system group (gid 0) most of the privileges traditionally restricted to the superuser. Only especially "dangerous" or "sensitive" operations are restricted to the superuser [Loucks 1986].

3. Other users/groups are allowed to perform remote directory/file mounts[7] if the process has search permission for the requested directory/file, owns the mounted upon object (directory/file) and has write permission in the parent directory of the mounted upon object.

The objectives of these constraints are to maintain system integrity but allowing users the flexibility to perform "casual" mounts. Userid/groupid translation, as discussed below, is implicit in the above definitions.

File System Implementation Issues

Virtual File Systems. The Distributed Services remote mount design uses the Virtual File System approach used with NFS [Sun 1986]. This approach allows construction of essentially arbitrary mount hierarchies, including mounting a local object over a remote object, mounting a remote object over a remote object, mounting an object more than once within the same hierarchy, mount hierarchies spanning more than one machine, etc. The main constraint is that mounts are only effective in the machine performing the mount.

Inherited mounts. It is desirable for one machine to be able to "inherit" mounts performed by other machines. For example, if a machine has mounted over `/usr/src/icon` and a second machine then mounts the first machine's `/usr/src`, it might be desired that the second machine see the mounted version of `/usr/src/icon`. This would not happen in the default case, but Distributed Services provides a query facility as part of a new `mntctl()` system call. The `mount` command supports a `-i` (inherited) flag which causes the query to be performed and the additional mounts to be made. By use of inherited mounts, clients of a file server need not know of restructuring of the server's mounts underneath the initial mount. For example, if a client always uses an inherited mount of `/usr/src`, it does not need to change it's configuration files when the server uses additional mounts to provide the subdirectories of `/usr/src`.

lookup. In conjunction with using the Virtual File System concept, we necessarily have replaced the traditional `namei()` kernel function, which translated a full path name to an i-number, with a component by component `lookup()` function. For file granularity mounts, the string form of the file name is used, along with the file handle of the (real) parent directory. This alternative to using the file handle for the mounted file allows replacement of the mounted file with a new version without loss of access to the file (with that name). (For example, when `/etc/passwd` is mounted and the `passwd` command is used, the old file is renamed `opasswd` and a new `passwd` file is produced. If we used a file handle for the file granularity mount, then the client would continue to access the old version of the file. Our approach gives the, presumably intended, effect that the client sees the new version of the file.)

Statelessness and Statefulness. One of the key implementation issues is the approach to "statelessness" and "statefulness." Wherever it is practical to use a stateless approach, we have done so. For example, our remote mounts are stateless. However, in some areas where we believe a stateful approach is necessary, we maintain state between server and client and are prepared to clean up this state information when a client or server fails. In particular, we maintain state with regard to directory and data caching, so that cache consistency can be assured.

Directory Caching. Use of component by component lookup means, in the worst case, that there will be a `lookup()` remote procedure call for each component of the path. To avoid this overhead in typical path searches, the results of `lookup()` calls are cached in kernel memory, for directory components only. Cached results may become invalid because of directory changes in the server. We believe that state information must be maintained for purposes of cache validity. Whenever any directory in a server is changed, client directory caches are purged. Only clients performing a

[7] Remote device mounts are not supported, but the only practical effect is that a remote device that is not mounted at all at the owning machine can not be remote mounted. This is likely desirable, since this situation is only likely to occur during maintenance of the unmounted device.

`lookup()` since the previous directory change are notified, and they, of course, only purge the entries for the server that had the directory change. This purpose of this strategy is to keep the directory cache entries correct, with little network traffic.

Data Caching. Distributed Services uses data caching in both client and server, to avoid unnecessary network traffic and associated delays. The caching achieves the traditional read ahead, write behind and reuse benefits associated with the kernel buffer cache, but with both client and server caches. As a result, read ahead (write behind) can be occurring in the client cache with regard to the network and in the server cache with regard to the disk. *As a result, disk to disk transfer rates to/from remote machines can be substantially greater than local rates.* In AIX we have carefully tuned the local disk subsystem, yet use of `cp` for remote files yields significantly higher disk to disk throughput than for local only files. Note that stateless designs may not support write behind, in order to guarantee that all data will be actually on the server's disk before the write rpc returns to the client.

Data Cache Consistency. In general, it is difficult to keep multiple cached data blocks consistent. We designed a general cache invalidation scheme, but chose to implement instead a state machine based on current opens of a given file. We emphasize that this mechanism is applied at a file granularity, and that it is strictly a performance optimization – the mechanism is designed to preserve the traditional multireader/multiwriter semantics of the UNIX file system. Any particular file will be in one of the following states:

1. Not open.
2. Open only on one machine This may be a different machine than the server for the file. ("async mode")
3. Open only for reads on more than one machine. ("read only mode")
4. Open on multiple machines, with at least one open for writing. ("fullsync mode")

We believe that the read only and async modes are dominant in actual system operation, and our client caching applies to these modes only. In fullsync mode, there is no client caching for the given file, but the server caches as in a standalone system.

Close/Reopen Optimization. A frequent scenario is that a file is closed, say by an editor, and then immediately reopened, say by a compiler. Our data cache consistency mechanisms are extended to allow reuse of cached data blocks in the client data cache, if and only if the file is not modified elsewhere between the close and subsequent reopen.

Kernel Structured Using Sun "vnode" Definition. We have used the Sun vnode data structure [Kleinman 1986] to support multiple file system types in the AIX kernel. This allows a clean division between the local AIX filesystem code and the remote filesystem code.

Virtual Circuit Interface. Distributed Services assumes virtual circuits are available for network traffic. One or more virtual circuits must remain in force between a client with a file open and the server for that file. (The mere existence of a remote mount does not require retention of a virtual circuit.) Execution of cleanup code, e.g., decrementing usage counts on open files, will be triggered by loss of a virtual circuit. The architecture of Distributed Services includes a *Virtual Circuit Interface* (VCI) layer to isolate the Distributed Services code from the supporting network code. Our current code uses the SNA LU 6.2 protocol to provide virtual circuit support, but, potentially, another connection oriented protocol, e.g., TCP, could be used. The basic primitives of the VCI are the `dsrpc()`, `dsrpc_got()` and `dsgetdata()` functions. `dsrpc()` acquires a connection with a specified machine and then issues `dsrpc_got()` to invoke a function on that machine. `dsrpc_got()` is called directly if the caller has a previously established connection available. Both of these calls return without waiting for the result of the remote function, allowing continued execution on the calling machine. `dsgetdata()` is used to request the result of a remote functions; it will wait until the result is available.

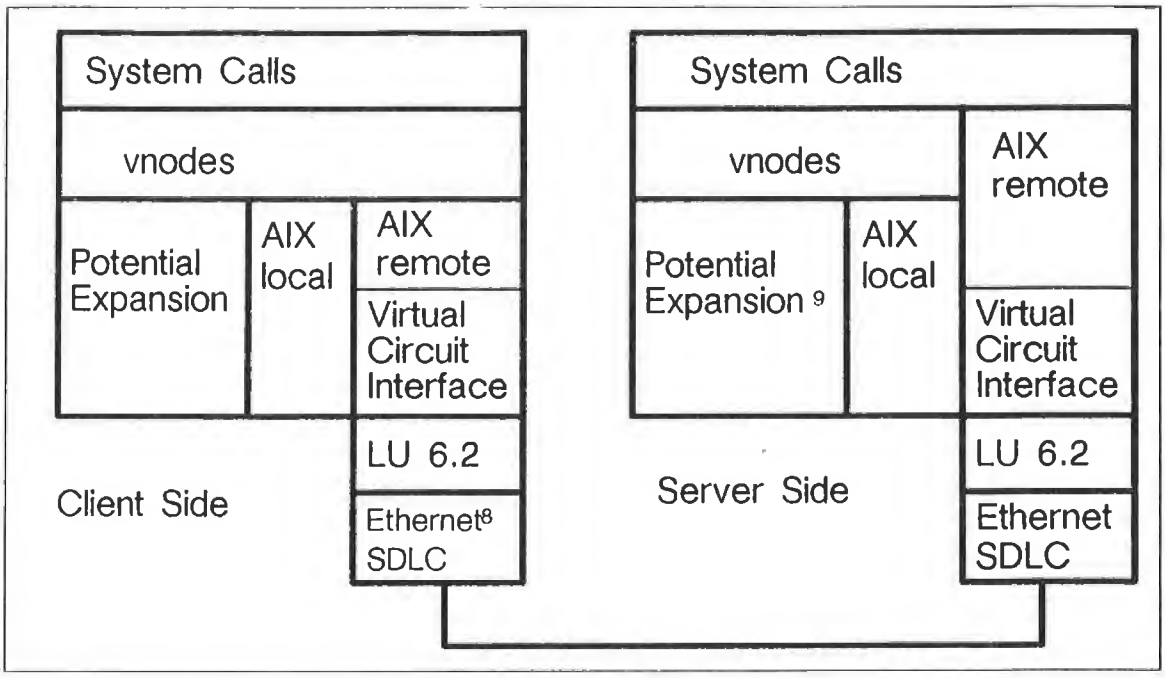

Figure 2. Architectural Structure of Distributed Services File System

SNA LU 6.2 Usage. We chose to use LU 6.2 because of its popular position in IBM's networking products and because of its technical advantages. In particular, LU 6.2 allows for "conversations" within a session. Conversations have the capabilities of virtual circuits, yet with low overhead of the order typically associated with datagrams. Typically, one or two sessions are opened to support the flow between two machines, regardless of the number of virtual circuits required. We have carefully tuned the LU 6.2 implementation, exploiting the fully preemptive process model of the AIX Virtual Resource Manager [Lang, Greenberg and Sauer 1986]. By properly exploiting the basic architecture of LU 6.2 and careful tuning, we have been able to achieve high performance without using special private protocols [Popek and Walker 1985] or limiting ourselves to datagrams.

The AIX implementation of LU 6.2 supports both Ethernet and SDLC transport. The AIX LU 6.2 and TCP/IP implementations are designed to coexist on the same Ethernet – in our development environment, we use both protocols on a single Ethernet, e.g., TCP for Telnet and/or X Windows and LU 6.2 for Distributed Services.

Distributed Services Security and Authorization

Encrypted Node Identification

When considering networks of the sort suggested by Figure 1, it is clear that each machine needs to be suspicious of the other machines. If a machine is going to act as a server for another, it should have a mechanism to determine that the potential client is not masquerading. The AIX implementation of SNA LU 6.2 provides an option for encrypted node identification between a pair of communicating machines. The identification is by exchange of DES encrypted messages. The identification occurs at session establishment time and at random intervals thereafter. Once a

[8] Ethernet is a trademark of Xerox Corporation.

[9] This is not intended as speculation of future products.

client/server have each determined that the other is not masquerading, then they can take appropriate actions authorized according to (the translated) userid's/groupid's associated with each request.

Userid/Groupid Translation

There are a number of reasons why a common userid space and a common group id space are impractical in the environment of Figure 1:

1. An individual machine, whether a private machine or a server, should not be required to give superuser (root) authority to a request from a process with root authority on another machine. Rather, it should be possible to reduce the authority of the remote process. The reduced authority may retain some administrative privileges, may be that of an ordinary user or may be no access at all, depending on the preferences of the administrator of the individual machine. Similar statements apply to the cluster of machines.
2. A user may have logins provided by several different administrators on several different machines/clusters, and these will typically have different numeric userids. When that user uses different machines, he/she should have access to his/her authorized resources on all machines in the network.
3. Previously operating machines may join a network or move to a new network, and existing networks may merge. When this happens, there may be different users/groups with the same numeric ids. Such reconfiguration should be possible without requiring users/groups to change numeric ids or changing userids/groupids in all of the inodes.

Our response to these requirements is to define a network wide ("wire") space of 32 bit userids and groupids. Each request leaving a machine has the userid translated to the wire userid and each request entering a machine has the wire userid translated to a local userid. The above requirements are met by proper management of the translations.

Distributed Services Administration

In addition to the normal system profiles, e.g., `/etc/filesystems`, there are profiles for both the SNA support and for Distributed Services. With these new profiles, we have taken care to organize the directories containing the profiles so that we can use remote mounts to administer remote machines, without use of remote login (or roller skates). For Distributed Services, there are three profiles, for machine ids and passwords, for userid/groupid translation and for registry of message queues.

Part of the AIX design is provision of a user interface architecture for a screen oriented ("menu") interface, to simplify system management and usage [Kilpatrick and Green 1986, Murphy and Verburg 1986]. Configuration of both SNA and Distributed Services, i.e., management of the SNA and Distributed Services profiles, is normally performed using menus conforming to this user interface architecture.

Distributed Services "Single System Image"

Our definition of "Single System Image" is as follows: *Users of the given system, users of external systems which communicate with the given system and application programmers ARE NOT aware of differences between single and multiple machine implementation. System administrators and maintenance personnel ARE aware of distinctions amongst machines.*

User/Programmer View of Distributed Services Single System Image

Though there are inherent exceptions to this, e.g., the `uname()` system call is designed to return the machine name, we believe that Distributed Services largely meets this definition. The key mechanism is to be able to properly configure the several machines so that they share the files and directories which matter to the user and the application programmer. These include basic profiles such as `/etc/passwd`, home directories, and directories containing applications, commands and libraries. Figure 3 sketches one such possible configuration.

Once this is accomplished, most of the desired properties just fall in place. The login process will be the same because of the sharing of `/etc/passwd` related files. Normal file system manipulations and applications work in the shared directories. Administrative commands for ordinary users, e.g., `passwd`, also work properly if they follow reasonable conventions (we had to rework several commands such as `passwd`, as discussed below.)

Administrator's View of Single System Image Configurations

Some of the administrator's tasks must be be performed for each machine individually. For example, the administrator must install and configure AIX and Distributed Services on each machine. Other tasks can be performed once for the entire single system image cluster. For example, installation of an application, in the usual case where the `installp` command retrieves files from diskette and places them in the appropriate subdirectory of `/usr/lpp`, need only be done once, assuming it is done after normal system startup. Similarly, the `adduser` command, which creates an entry in `/etc/passwd`, creates a home directory and copies standard files to the home directory, need only be applied once.

Routine maintenance, e.g., backing up and restoring files, can be done for the system as a whole while the system is in normal operation. Error logs are intentionally kept separately for each machine – otherwise, the first problem determination step would be to isolate the anomalous machine. Some maintenance operations, e.g., image backups of disks and hardware diagnostics, are necessarily performed on a machine by machine basis, while the machine is in maintenance mode.

Implementation Issues in Distributed Services Single System Image

There is an obvious question of ordering in starting the separate machines. We have added a number of options to the `mount` command and `/etc/filesystems` to allow simple retry mechanisms to be executed in the background when initial mount attempts fail. This is done to allow arbitrary ordering of the startup of machines.

Many of the interesting commands, e.g., `passwd`, use private locking mechanisms, e.g., based on creating/deleting dummy lock files. We have had to modify a number of these commands to use the `lockf()` system call.

A more subtle issue is the "copy/modify/unlink/relink" idiom used in a number of interesting programs such as editors. This idiom does not work in all cases of file granularity mounts, because a client may be attempting to violate the prohibition of linking across devices. In more detail, the idiom is as follows, for updating `foo` in the current directory:

1. `cp foo .foo.tmp`
2. modify `.foo.tmp`
3. `rm foo`
4. `ln .foo.tmp foo`
5. `rm .foo.tmp`

If `foo` is a file mount from a different device, step 4 will fail. We have had to modify several programs to do a copy if the link step (4) fails. Note that this is not a problem with directory mounts, only file granularity mounts.

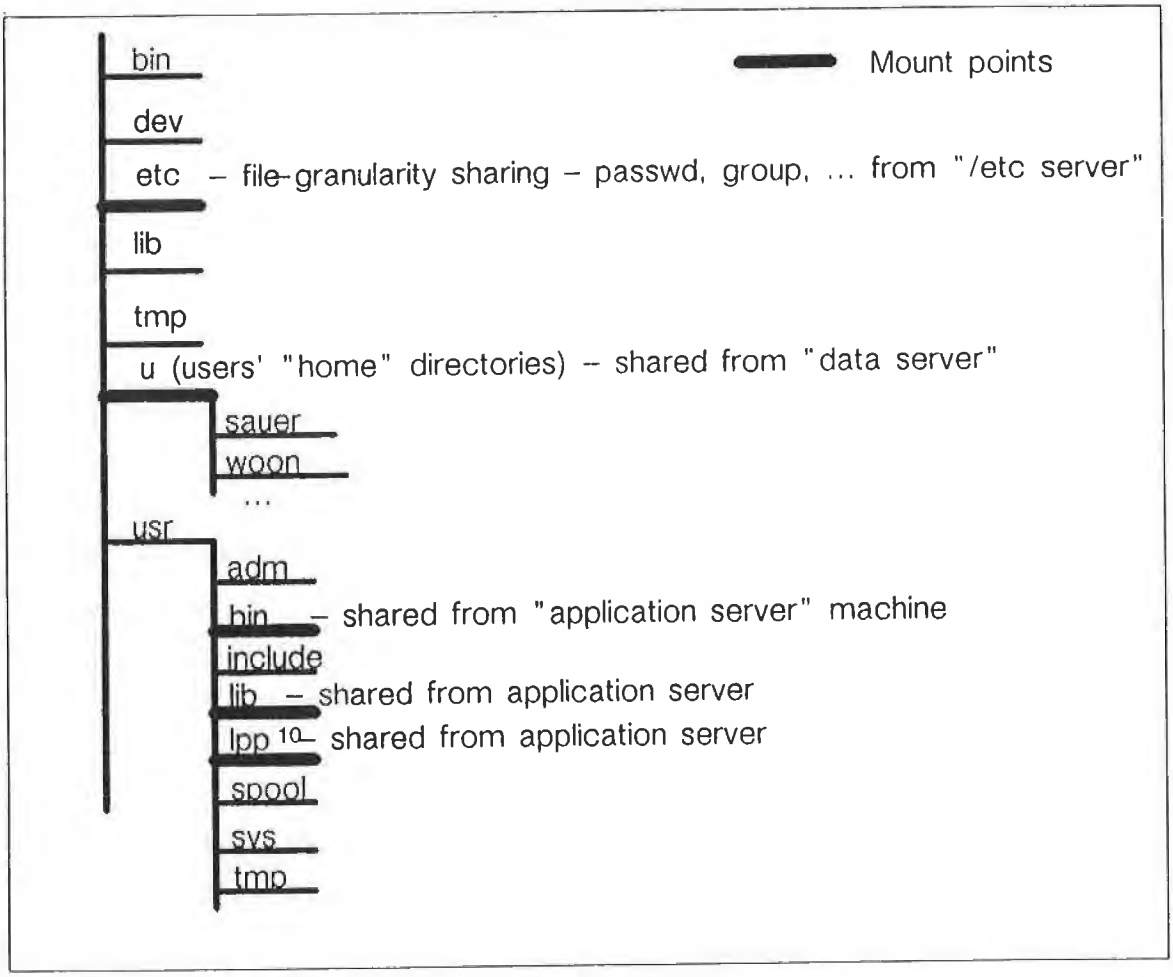

Figure 3. Example Shared File System.

There is also a potential problem with routines such as `mktemp()` and `tempnam()`, which use process ids to generate unique file names. Since process ids are not unique across machines, we have modified these routines to use the machine id as well as the process id in deriving a file name. (The modified versions of these routines are packaged with AIX, so that object code does not have to be recompiled/relinked to run with Distributed Services.)

Separate Machine Operation

Clearly, it is desirable that a client machine of the servers in Figure 3 be able to operate if one or more of the servers is down. A critical aspect of this is having recent copies of the shared files from the "`/etc` server." As part of the mounting of these files, before the mount is actually performed, the file is copied from the server to the client. For example, before mounting the shared `/etc/passwd` over the client `/etc/passwd`, the shared version is mounted temporarily over another file and copied to `/etc/passwd`. For each user that is to be able to use a machine when the "home directory server" is not available, a home directory must be created and stocked with essential data files. Similarly, for a machine to be able to use an application when the "application server" is not available, that application must be installed in the client's `/usr/lpp`, when the server's `/usr/lpp` is

[10] An AIX convention is to place most applications in subdirectories of `/usr/lpp`.

not mounted. The resulting machine is certainly not as useful as when the servers are available, but it is usable, and much better than no machine at all.

Summary

We believe we have done well in meeting our design goals:

1. Distributed Services provides local/remote transparency for ordinary files (both data and programs), for directories and for message queues.
2. Our implementation adheres closely to AIX semantics, except for the lack of support for remote mapped files.
3. We have achieved good remote performance in general, and some remote operations are actually faster than corresponding local operations.
4. Use of a popular network protocol, SNA LU 6.2, gives us synergy with other SNA development and independence of the underlying transport media.
5. We have been careful to provide for flexibility in configurations and administrative environments.
6. Our encrypted node identification and id translation mechanisms give us strong control over security and authorization.
7. Our use of architectures such as LU 6.2, the vnode concept, our Virtual Circuit Interface, etc. allows us substantial room for potential extension and growth in network media, file systems and network protocols, respectively.

Further, we believe we have advanced the state of the art with the following

1. Our simple, but effective approach to single system image.
2. Use of a standard virtual circuit protocol, SNA LU 6.2, while achieving high performance.
3. Our performance optimizations, especially our caching strategies.
4. Our extensions for administrative flexibility and control, e.g., file granularity mounts, inherited mounts, administration based on remote mounting of profiles, etc.

References

1. IBM, *IBM RT Personal Computer AIX Operating System Technical Reference Manual*, SA23-0806, January 1986.
2. P. J. Kilpatrick and C. Greene, "Restructuring the AIX User Interface," *IBM RT Personal Computer Technology*, SA23-1057, January 1986.
3. S. R. Kleinman, "Vnodes: An Architecture for Multiple File System Types in Sun UNIX," *USENIX Conference Proceedings,* Atlanta, June 1986.
4. T. G. Lang, M. S. Greenberg and C. H. Sauer, "The Virtual Resource Manager," *IBM RT Personal Computer Technology*, SA23-1057, January 1986.
5. L. K. Loucks, "IBM RT PC AIX Kernel – Modifications and Extensions," *IBM RT Personal Computer Technology*, SA23-1057, January 1986.
6. T. Murphy and R. Verburg, "Extendable High-Level AIX User Interface," *IBM RT Personal Computer Technology,* SA23-1057, January 1986.
7. G. Popek, B. Walker, J. Chow, D. Edwards, C. Kline, G. Rudisin and G. Thiel, "A Network Transparent, High Reliability Distributed System," *Proceedings of the 8th Symposium on Operating Systems Principles,* Pacific Grove, CA, 1981.

8. G. Popek and B. Walker, *The LOCUS Distributed Operating System,* MIT Press, 1985.

9. A. P. Rifkin, M. P. Forbes, Richard L. Hamilton, M. Sabrio, S. Shah and K. Yueh, "RFS Architectural Overview," *USENIX Conference Proceedings,* Atlanta, June 1986.

10. R. Sandberg, D. Goldberg, S. Kleinman, Dan Walsh and B. Lyon, "Design and Implementation of the Sun Network File System," *USENIX Conference Proceedings,* Portland, June 1985.

11. Sun Microsystems, Inc., *Networking on the Sun Workstation,* February 1986.

= = =

The 1988 Election of the USENIX Board of Directors

This is to remind all USENIX members of the 1988 elections. The Nominating Committee was announced in two previous issues of *;login:*. The nominees will be announced at the 1988 Dallas Technical Conference, the first week of February.

The following have agreed to serve on the Committee:

Lew Law (chair)	{usenix,harvard}!law 617-495-2627
Bruce Borden	hplabs!dana!bsb 415-960-1980
Tom Ferrin	tef@cgl.ucsf.edu 415-476-1100
Mike O'Dell	mo@seismo 703-893-3660
Peter Langston	psl@bellcore 206-641-6106
Mike Tilson	mike@hcrvax 416-922-1937

The Board is comprised of eight members: President, Vice-President, Treasurer, Secretary, and four Board members.

Some attributes which contribute to promoting the smooth functioning and continuing development of the Association are:

- a real interest in what the USENIX Association is doing,
- a committment to improve and expand the Association activities,
- willingness and available time to commit to Association activities.

If you are interested, know of anyone who is or might be interested, or know of anyone who would be an asset to the USENIX Association as a Board member, and willing to serve, please contact me or any member of the Nominating Committee.

Further, the bylaws state that "Nominations ... may also be made by any five members. All nominations must bear the signature of at least five Voting Members."

Peter H. Salus
Executive Director

Book Reviews

UNIX System Security

by Patrick H. Wood and Stephen G. Kochan
(Hayden Book Company) 299 Pages, $34.95

Reviewed by Robert E. Van Cleef

NAS-RNS Workstation Subsystem Manager
General Electric Corporation
NASA Ames Research Center
Mail Stop 258-6
Moffet Field, CA 94035

Summary: A very valuable book. A must read for any UNIX system administrator.

With the current proliferation of UNIX systems in areas outside of the traditional University and Research environments, UNIX system security is becoming a real concern for many people. Unfortunately, most of the currently available books on the UNIX operating system avoid any discussion of security beyond using user passwords and basic file protection. Current system manuals only discuss bits and pieces of system security in a haphazard fashion. This means that anyone wanting to evaluate their system's security has a problem. In fact, even if you are an experienced computer user, if you come from a non-UNIX environment you will not be able to find the information you need in the manuals that are being delivered with many of the newer UNIX boxes, including AT&T's.

Many of us "old-timers" were first introduced to UNIX security concerns by reading D. M. Ritchie's [1] and R. H. Morris' [2] discussions on system security in Volume Two of the UNIX Programmer's Manual, but most vendors don't deliver Volume Two with their systems any more. And I wonder how many people in the small system world would have copies of Ken Thompson's Turing Award lecture, on Trusting Trust [3] in their personal libraries? Knowledge of UNIX system security can be hard to come by.

Seeing this need, Patrick Wood and Stephen Kochan have collected all of the bits and pieces into a concise and very readable reference manual that should be on the shelf of every UNIX system administrator. Their stated goal was to "teach security awareness to UNIX users and administrators." They not only achieved that goal, but they have provided a collection of tools to help the system administrator determine the level of security of their system, and to alter that level of security to meet their needs. They achieve this with a high level of readability and accuracy. I found it almost impossible to put down – highly unusual for a technical book. There was also a side affect in that some of their sample programs and shell scripts taught me some very useful "tricks."

This is not just another introduction to UNIX book. The authors assume that you understand enough about shell scripting and C programs to allow them to concentrate on their subject and not spend their time teaching programming. On the other hand, they do explain completely, and with many examples, the shell scripts and C programs that they introduce. In fact, in some ways this can be considered a "Cook Book" on system security, in that they include appendixes with the full source listings for the programs that they discuss. But it is more than just a cook book: like most good security manuals it not only shows you what security holes are in you system, but it gives you the tools to close those holes, with a complete index to allow you to find those examples when you need them.

Finally, a thought to those of you who aren't worried about security. If you don't pay

attention to the security holes discussed in this book, someone else might . . .

Chapter 1, Introduction.

Chapter 2, A Perspective on Security, is the complete written testimony of Robert Morris, of AT&T's Bell Laboratory, before the House Committee on Science and Technology's subcommittee on Transportation, Aviation and Materials.

Chapter 3, Security for Users, not only covers the traditional subjects, such as password security, file permissions, Set User ID and Set Group ID programs, it also discusses the day-to-day security problems associated with such common programs as *cp, mv, ln,* and *cpio*. The discussion includes Trojan horses, viruses, and the problems with Intelligent Terminals, and how to deal with them.

Chapter 4, Security for Programmers, introduces you to the writing of secure programs. They discuss process control, file attributes, UID and GID processing, and dealing with the password file. They give you clear guidelines for SUID/SGID programs, and show you how to write a SUID/SGID program to safely allow selective access to protected data using both a shell script and a C program.

Chapter 5, Security for Administrators, starts with system file and device file permissions. In this chapter the authors put together something that I have never seen elsewhere, a full discussion of the administrator's security responsibilities, complete with a discussion on the proper use of the Super User privileges. They discuss password aging and control, and introduce the use of restricted shells. Under a summation on small system security, they also discussion the physical security of the system.

Chapter 6, Network Security, mainly discusses the UUCP world, including the Honeydanber UUCP that now is part of System V, but it does include some discussion of the problems associated with RJE links, NSC's Hyperchannel network, and AT&T's 3B Net, and using encrypted data links.

Appendix A – References.

Appendix B – Security Commands and Functions.

Appendix C – Permissions.

Appendix D – Security Auditing Program. *secure* – perform a security audit on a UNIX system.

Appendix E – File Permission Program. *perms* – check and set file permissions.

Appendix F – Password Administration Program. *pwadm* – perform password aging administration.

Appendix G – Password Expiration Program. *pwexp* – prints weeks to expiration of user's password.

Appendix H – Terminal Securing Program. *lock* – locks terminal until correct password is entered.

Appendix I – SUID/SGID Shell Execution Program. *setsh* – run a SUID/SGID/execute-only shell.

Appendix J – Restricted Environment Program. *restrict* – establishes a user in a restricted environment.

Appendix K – DES Encryption Program. *descrypt* – encode/decode using DES.

Appendix L – SUID Patent. A copy of US Patent 4,135,240, D. M. Ritchie's patent of the SUID concept.

Appendix M – Glossary

Index

References

[1] D. M. Ritchie, "On the Security of UNIX," UNIX Programmer's Manual, Section 2, AT&T Bell Laboratories.

[2] R. H. Morris and K. Thompson, "Password Security: A Case History," UNIX Programmer's Manual, Section 2, AT&T Bell Laboratories.

[3] K. Thompson, "Reflections On Trusting Trust," 1983 ACM Turing Award Lecture, CACM, Volume 27, Number 8 (August 1984), pp. 761-763.

troff typesetting for UNIX Systems

by Sandra L. Emerson and Karen Paulsell
(Prentice Hall, 1987, ISBN 0-13-930959-4)

Reviewed by Jaap Akkerhuis

CWI, Amsterdam
mcvax!nl.cwi!jaap

Goal of the Book

The book is intended to be an introduction to the use of troff for the novice and also a reference manual for experienced users. It tries to correct the lack of adequate end-user documentation for troff. Alas, any explanation about the concepts of troff –or any other formatting program is missing. For instance, the term "partial collected lines" is used a lot but never explained.

As an introduction to troff the authors explain all the basic requests and how to write macros. It is a pity that they do so in a haphazard way. They often use a request, like `.de`, with the remark that the full details will be explained further on in the book. This is sometimes confusing. Apparently, the authors did not have a clear idea on how to introduce a novice to the game of troff.

What I do like is that they give a full treatment of the `.nx` and `.rd` requests. Hardly any of the existing literature explains the possibilities of creating form letters with n/troff using these requests. Also, every possible troff request is explained, each description accompanied with an example of its use. But for the more experienced user there is not a lot new. Even small tricks, for example, what you can do with the `.ss` request, are not explained. Fancy techniques, like how to do balanced columns, are not handled at all. The chapters about the preprocessors and macro packages are sketchy and don't give more information than the existing literature.

To be a reference manual, it should at least replace the original n/troff reference manual. Some finer points haven't been covered, like the full definition of certain requests, for instance, the append to macro command: `.am xx yy`. So, don't throw the original manual from Ossanna out of the window; you will still need it.

Typesetting

The most disturbing and misleading thing about the book is its title. Apart from a remark like "You should think as a typesetter," there is nothing in the book about typesetting or the noble art of typography. All the examples deal with the standard non-interesting cases of typesetting.

The typesetting of the book itself is not really done exceptionally well, it is just another book which is typeset by the authors. I'm always wondering why authors don't ask advice from a typographical consultant, it would do miracles for book design. Of course, this is partly the failure of the publisher. These firms are more and more interested in making money by cranking out printed paper and not caring at all about how the product looks. I'm afraid that ignoring the issues involved with typography in this book will lead to even more horrible looking books than there are around already.

Errors in the Book

In general, there will always be errors in books. In this case, the advanced troff user will spot them easily, but for the novice they may be very disturbing.

The first one pops up in the first example in the first chapter (pages 3 & 4). This one can be waved away if you consider that novice shouldn't be hampered too much with details, but the next example (page 5) is unforgivable. The quoted troff source of `.PP` for the *-ms* macro package is missing some back slashes! This demonstrates again that it is not always easy to write about a tool by using it. There are more places in the book where these things happen. When showing the pitfalls of the arithmetic in troff using the `.ll` request the complete promised test file isn't around. Some parts of how the file might have looked and some of the (incorrect) output is shown. Something really went wrong there.

Who Should Buy the Book

Although I'm not very impressed by the book, it may be of some use for a lot of people. There are many UNIX systems around which don't provide the original documentation. For these cases, the book fills a gap. Also, people complaining about the terseness of the original reference manual might want to read it.

= = =

COMPUTING SYSTEMS – New USENIX Quarterly

The USENIX Association is pleased to announce that beginning early in 1988 it will be publishing a quarterly

COMPUTING SYSTEMS
The USENIX Association Journal

Michael D. O'Dell, Maxim Technologies, will serve as Editor-in-Chief.

This journal will be dedicated to the analysis and understanding of the theory, art, design, engineering, and implementation of advanced computing systems, with an emphasis on systems inspired or influenced by the UNIX tradition. Articles concerning operating systems, architecture, networking, programming languages, and sophisticated applications are of interest. Papers will reflect a mix of theory and practical experience.

Submissions, in *nroff/troff* format should be sent to *{uunet,ucbvax,...}!usenix!journal*. Hard copy submissions should be supplied in five (5) copies and mailed to

Computing Systems
USENIX Association
P.O. Box 2299
Berkeley, CA 94710

The Association hopes to have a rapid turnaround, with only 6-8 months between submission and publication.

Work-in-Progress Reports from the Phoenix Conference

The Siemens RTL Tiled Window Manager

Ellis S. Cohen
Mark R. Biggers
Joseph C. Camaratta

Siemens Research & Technology Laboratories
105 College Road East
Princeton NJ 08540-6668
(609) 734-6524
siemens!ellis (uucp)
ellis.cohen@a.gp.cs.cmu.edu (arpa)

The Siemens RTL Tiled Window Manager is a network window manager which is currently implemented on a Sun and interacts with clients using the CMU/ITC Andrew protocol. It has been in operation since Fall 1986, and is the window manager of choice within our group. It is currently being ported to run as a window manager for X.

Except for menus and transient pop-up windows, a tiled window manager does not permit windows to overlap. If opening, moving, or enlarging a window would cause overlap, then either the operation is disallowed, or windows are automatically shrunk, moved, or closed to avoid the overlap.

Unlike other tiled window managers which lay out windows in columns, the Siemens RTL window manager supports arbitrary tiled layouts and thus avoids the limitations of other systems. The system is also distinguished by a number of important features including:

- Minimum sizes to better support what appear as "slivers" in overlapping systems – the portion of a window which remains partially visible while other windows are fully visible.
- Desired sizes – a preferred size of the window (as set by the user) which the system automatically attempts to maintain. A window may automatically be enlarged to its desired size when it becomes the focus – analogous to exposing a window upon focus in an overlapping system.
- Zooming as both a means of temporarily enlarging a window and icons to represent windows which are closed.
- Automatic placement for finding the best place to open a window.
- Automatic prorating for fairly allocating space when not all windows on the screen can attain their desired size as well as automatic filling for fairly allocating additional space to windows when extra space is available on the screen.
- Automatic plowing for shrinking or moving windows out of the way when a window is opened or enlarged. Plowing is used instead of prorating when the goal is to cause changes in as few adjacent windows as possible.
- Enlargement for making a window as large as possible by shrinking, but not closing, other windows on the screen.
- The use of dynamically resettable options (initialized via profiles) for controlling the degree of automation and the character of the user interface.

Camelot: A Full Function, Distributed Transaction Facility for the MACH, BSD 4.3-Compatible Operating System

Jeffrey L. Eppinger

Department of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
(arpanet: jle@spice.cs.cmu.edu)

The Camelot distributed transaction facility is designed to simplify the construction of reliable, distributed applications that access shared data. Camelot provides simple interfaces (via macros and C library calls) for executing transactions and defining servers that encapsulate permanent data objects. Camelot runs on the Mach, BSD 4.3-compatible operating system and should

execute on all the uniprocessors and multiprocessors that Mach supports. Implementations of Camelot for the RT PC and VAX computers are currently being tested. Some work remains before the system can be released to others. Josh Bloch, Dean Daniels, Rich Draves, Dan Duchamp, Sherri Menees, Alfred Spector, Dean Thompson, and the speaker have designed and built Camelot.

This presentation will briefly describe Camelot and its status. In particular, it will describe Camelot's goals, features, programming interface (i.e., man 3 library interface), implementation, and performance. Camelot will demonstrate that transaction processing is an easy to use, useful programming technology that can be very efficiently implemented for the Mach environment. A user's guide and other documentation can be obtained from the speaker.

Authentication for Untrusted Networks of Untrusted Hosts

Dan Geer

Manager of Systems Development
Project Athena - E40-342F
Massachusetts Institute of Technology
1 Amherst Street
Cambridge, Massachusetts 02139
geer@athena.mit.edu

Most conventional time-sharing systems require a prospective user to identify him or herself and to authenticate that identity before using its services. In an environment consisting of a network that connects prospective clients with services, a network service has a corresponding need to identify and authenticate its clients. When the client is a user of a time-sharing system, one approach is for the service to trust the authentication that was performed by the time-sharing system. For example, the network applications lpr and rcp provided with Berkeley 4.3 UNIX trust the user's time-sharing system to reliably authenticate its clients.

In contrast with the time-sharing system, in which a protection wall separates the operating system from its users, a workstation is under the complete control of its user, to the extent that the user can run a private version of the operating system, or even replace the machine itself. As a result, a network service cannot rely on the integrity of the workstation operating system when it (the network service) performs authentication.

The Kerberos design extends the conventional notions of authentication, authorization and accounting to the network environment with untrusted workstations. It establishes a trusted third-party service, Kerberos, that can perform authentication to the mutual satisfaction of both clients and services. The authentication approach allows for integration with authorization and accounting facilities. The resulting design is also applicable to a mixed time-sharing / network environment in which a network service is not willing to rely on the authentication performed by the client's time-sharing system.

The Kerberos system will be made available in a manner consistent with our previous release of the X-Window System. Parties with serious interest in participating in a beta-test of this software are invited to contact Athena at this time.

Threads in System V: Letting UNIX Parallel Process

J. C. Wagner
J. M. Barton

Silicon Graphics, Incorporated

With the rise of numerous multiprocessor UNIX machines, programming environments that allow easy use of the newly available power are becoming more important. At SGI, considerable effort is being placed on the design and implementation of a powerful parallel programming environment.

Among the enhancements under development are a multi-threaded execution process model, expanded shared memory features, user-level synchronizing features and matching debugging capabilities. The main goals for the multi-threaded processes is to offer independently schedulable threads with a

create/destroy rate an order of magnitude faster than the typical *fork*(2) system call and context switch times significantly faster than process to process. This allows threads to be used for both classic cooperating task applications as well as parallelizing compilers. Threads share all text and data of the underlying process and have the same UNIX attributes as the underlying process – one pid, one user area, same file table, etc., implying that any instance of a thread can execute inside the kernel. Other enhancements include *ptrace* support for retrieving task state information. This is combined with a graphical debugger to allow multiple threads to be displayed and debugged simultaneously. The System V shared memory facility is enhanced to allow heap allocation out of shared segments, growable segments, and system cleanup of unused segments. Programs that need multiple threads performing system calls can then simply use *fork*(2), and use shared memory as the current data space; this gives functionality superior to many system's "shared fork."

The multi-thread work is being integrated into a stable multi-processor System V kernel. The low level process/thread management and scheduling code has has been running for 2 months. The base set of kernel synchronization primitives have been implemented. Multiple tasks in a process have been working for 1 month, current performance tests show it to be ~10× the system fork rate. The kernel hooks for accessing the state information of tasks is also in place. The continued effort involves performance enhancements, full scale user synchronization primitives, debugger integration, and inter-thread messaging.

CCMD: A Version of COMND in C

Andrew Lowry
Howard Kaye

Columbia University

CCMD is a general parsing mechanism for developing User Interfaces to programs. It is based on the functionality of TOPS-20's COMND Jsys. CCMD allows a program to parse for various field types (file names, user names, dates and times, keywords, numbers, arbitrary text, tokens, etc.). It is meant to supply a homogeneous user interface across a variety of machines and operating systems for C programs. It currently runs under System V UNIX, 4.2/4.3 BSD, Ultrix 1.2/2.0, and MS-DOS. The library defines various default actions (user settable), and allows field completion, help, file indirection, comments, etc. on a per field basis. Future plans include command line editing, command history, and ports to other operating systems (such as VMS).

CCMD is available for anonymous FTP from

[CU20B.COLUMBIA.EDU]WS:<SOURCE.CCMD>*.*

For further information, send mail to:

info-ccmd-request@cu20b.columbia.edu
seismo!columbia!cunixc!info-ccmd-request

CAP – Columbia AppleTalk Package for UNIX (4.2 BSD)

(For use with AppleTalk/Ethernet bridge)

Charlie C. Kim

User Services Group
Libraries and Center for Computing Activities

Bill Schilit

Columbia University

CAP is written under UNIX BSD 4.2 and implements a portion of Apple Computer's AppleTalk protocols. In order to use this package you need an AppleTalk/Ethernet bridge (e.g. Kinetics FastPath box).

CAP routines are structured, for the most part, the same as the Apple routines described in "Inside AppleTalk" and "Inside LaserWriter." Refer to the Apple documents and the procedure comments for a complete description of the routines and how to call them.

Bill Croft's original work in this area provided the inspiration for CAP.

Availability

Release

Distribution at present is limited to anonymous FTP from CU20B.COLUMBIA.EDU. See the Release notes in the directory US:<US.CCK.CAP.D4> in the file RELEASE for information on the current release (including the file locations). There are no current plans for any other types of distribution.

MDPIC – MacDraw to PIC Converter

Daniel Klein

Avatar Corp.
5606 Northumberland
Pittsburgh, PA 15217
412/422-0285

PIC provides users of UNIX a convenient, but somewhat difficult mechanism for drawing pictures inside of troff documents. All of PIC's commands are in "plain English," but it is usually very difficult to envision the pictoral representation of a set of words. The Macintosh, on the other hand, provides users a simple to use, easy to visualize WYSIWYG (what you see is what you get) drawing mechanism. Unfortunately, the only way to interface a Macintosh with *troff* was through physical cut and paste. No automated, electronic mechanism existed.

A new program, (MDPIC, for MacDraw to PIC) has been developed, that allows users to create intricate pictures with a Macintosh, and include them directly into a *troff* document. The MDPIC program takes a binary representation of the MacDraw picture (confusingly called PICT format), and translates this picture into UNIX PIC format. Most of the Macintosh drawing primitives are translated accurately – boxes, rounded rectangles, arcs, ovals, circles, arrows, text, and lines. Fill patterns are supported, although a 100% accurate representation is not possible (the Macintosh can draw with arbitrarily colored pens, while PIC supports only a black pen). Varying line sizes, fonts, point sizes, and dotted lines are all translated accurately. In short, anything that would ordinarily be drawn with PIC can be accurately translated from the Macintosh.

MDPIC enables creators of documents to quickly generate pictures to include in their documents using a WYSIWYG editor, then translate them to a format usable by UNIX. Since this new format is human readable, changes to the pictures may either be made in the original WYSIWYG form, or in the PIC source file that MDPIC generates. Because the Macintosh presents all of its pictures graphically, the user is spared the headache of painstakingly measuring and placing all of the lines, arcs, and boxes in a picture – a task that is required by using PIC alone.

MDPIC presently runs on the Sun, VAX, and Masscomp machines, and a future port directly to the Macintosh is planned for the future.

Directional Selection is Easy as Pie Menus!

Don Hopkins

University of Maryland
Heterogeneous Systems Laboratory
College Park, MD 20742
(301) 454-1516

Simple Simon popped a Pie Men-
-u upon the screen;
With directional selection,
all is peachy keen!

The choices of a Pie Menu are positioned in a circle around the cursor, instead of in a linear row or column. The choice regions are shaped like the slices of a pie. The cursor begins in the center of the menu, in an inactive region that makes no selection. The target areas are all adjacent to the cursor, but in a different directions.

Cursor direction defines the choice. The distance from the menu center to the cursor, because it's independent of the direction, may serve to modify the choice. The further away from the Pie Menu center the cursor is, the more precise the control of the selection is, as the Pie slice widens with distance.

With familiar menus, choices can be made without even seeing the menu, because it's the direction, not the distance, that's important. "Mousing ahead" with Pie Menus is very easy and reliable. Experienced users can make selections quickly enough that it is not actually necessary to display the menu on the screen, if the mouse clicks that would determine the selection are already in the input queue.

The circular arrangement of Pie Menu items is quite appropriate for certain tasks, such as inputing hours, minutes, seconds, angles, and directions. Choices may be placed in intuitive, mnemonic directions, with opposite choices across from each other, orthogonal pairs at right angles, and other appropriate arrangements.

Pie menus have been implemented for uwm, a window manager for X-Windows version 10, for the SunView window system, and for NeWS, Sun's extensible PostScript window system. Don Hopkins did the uwm and NeWS implementations, and Mark Weiser did the SunView implementation.

Jack Callahan has shown Pie Menus to be faster and more reliable than linear menus, in a controlled experiment using subjects with little or no mouse experience. Three types of eight-item menu task groupings were used: Pie tasks (North, NE, East, etc...), linear tasks (First, Second, Third, etc...), and unclassified tasks (Center, Bold, Italic, etc...). Subjects were presented menus in both linear and Pie formats, and told to make a certain selection from each. They were able to make selections 15% faster, with fewer errors, for all three task groupings, using Pie Menus. Ben Shneiderman gave advice on the design of the experiment, and Don Hopkins implemented it in Forth and C, on top of the X-Windows uwm.

The disadvantage of Pie Menus is that they generally take up more area on the screen than linear menus. However, the extra area does participate in the selection. The wedge-shaped choice regions do not have to end at the edge of the menu window – they may extend out to the screen edge, so that the menu window only needs to be big enough to hold the choice labels.

Proper handling of pop-up Pie Menus near the screen edge is important. The menu should idealy be centered at the point where the cursor was when the mouse button was pressed. If the menu must be moved a certain amount from its ideal location, so that it fits entirely on the screen, then the cursor should be "warped" by that same amount.

Pie Menus encompass most uses of linear menus, while introducing many more, because of their extra dimension. They can be used with various types of input devices, such as mice, touch pads, graphics tablets, joysticks, light pens, arrow keypads, and eye motion sensors. They provide a practical, intuitive, efficient way of making selections that is quick and easy to learn. And best of all, they are not proprietary, patented, or restricted in any way, so take a look and feel free!

References:

"Pies: Implementation, Evaluation, and Application of Circular Menus," By Don Hopkins, Jack Callahan, and Mark Weiser (Paper in preparation. Draft available from authors.)

"A Comparative Analysis of Pie Menu Performance," By Jack Callahan, Don Hopkins, Mark Weiser, and Ben Shneiderman (Paper in preparation. Draft available from authors.)

Worldnet – Computer Networks Worldwide

John S. Quarterman

Texas Internet Consulting
701 Brazos, Suite 500
Austin, TX 78701-3243

(512) 320-9031
jsq@longway.tic.com
uunet!longway!jsq

Computer networks extend throughout the world, have millions of users, and provide unique services. Yet the only publication that has dealt with existing networks at length on a global scale is my article "Notable Computer Networks" in the October 1986 *Communications of the ACM* [Quarterman1986] (the longest article CACM has ever published and one well-received by the readers). I propose to write a book on the same subject to include material which could not be fit into the article. Ths book will discuss existing computer networks throughout the world, their interconnections, the services they provide, their composition, their administration, their users, and their effects. The article is already 40 pages (30,000 words) in CACM, or about 60 book pages. There is at least that much more textual information available, and there will be many pages of additional tables, figures, and maps, so the book should be about 200 to 250 pages long.

Reference:

[Quarterman1986] John S. Quarterman and Josiah C. Hoskins, "Notable Computer Networks," *Communications of the ACM*, vol. 29, no. 10, pp. 932-971, Association for Computing Machinery, New York, NY, October 1986.

Multiple Programs in One UNIX Process

Don Libes

[The full report appeared on pages 7-13 of the July/August 1987 issue of ;login:.]

Have You M-o-v-e-d?

Summer seems to be a time when many folks move or change jobs. The result is that we may not have your correct address.

If this issue of *;login:* was addressed incorrectly, **PLEASE** fill out the "change of address" form on the back of the issue and send it in to us.

If you have been getting more than one copy of the Workshop or Conference announcements, please let us know (sending back the "extra" envelopes or labels would help).

Future Meetings

4th Computer Graphics Workshop
Oct. 8 & 9, 1987, Cambridge, MA

* * *

POSIX Portability Workshop
Oct. 22 & 23, 1987, Berkeley, CA

* * *

C++ Workshop
Nov. 9 & 10, 1987, Santa Fe, NM

* * *

USENIX 1988 Winter Conference and UniForum – Dallas

The USENIX 1988 Winter Conference, featuring tutorials and technical sessions, will be held on February 9-12, 1988, at the Grand Kempinski Hotel in Dallas, Texas. It will be concurrent with UniForum 1988, which will also be in Dallas.

USENIX 1988 Summer Conference and Exhibition – San Francisco

The USENIX 1988 Summer Conference and Exhibition will be held on June 20-24, 1988, at the Hilton Hotel in San Francisco, California. There will be a conference, tutorials, and vendor exhibits.

Long-term USENIX Conference Schedule

Feb 8-12	'88 Grand Kempinski, Dallas
Jun 20-24	'88 Hilton Hotel, San Francisco
Jan 31-Feb 3	'89 Town & Country Inn, San Diego
Jun 12-16	'89 Hyatt Regency, Baltimore
Jan 23-26	'90 Washington, DC
Jun 11-15	'90 Marriott Hotel, Anaheim
Jan 22-25	'91 Dallas
Jun 10-14	'91 Opryland, Nashville

= = =

Publications Available

The following publications are available from the Association Office. Prices and overseas postage charges are per copy. California residents please add applicable sales tax. Payments **must** be enclosed with the order and **must** be in US dollars payable on a US bank.

The EUUG Newsletter, which is published four times a year, is available for $4 per copy or $16 for a full-year subscription.

The July 1983 edition of the EUUG Micros Catalog is available for $8 per copy.

Conference and Workshop Proceedings

				Overseas Mail		
Meeting	Location	Date	Price	Air	Surface	Source
USENIX	Phoenix	Summer '87	$20	$25	$5	USENIX
USENIX	Wash. DC	Winter '87	$20	$25	$5	USENIX
Graphics Workshop III	Monterey	December '86	$10	$15	$5	USENIX
USENIX	Atlanta	Summer '86	$10	$25	$5	USENIX
USENIX	Denver	Winter '86	$10	$25	$5	USENIX
Graphics Workshop II	Monterey	December '85	$ 3	$ 7	$5	USENIX
USENIX	Dallas	Winter '85	$10	$25	$5	USENIX
Graphics Workshop I	Monterey	December '84	$ 3	$ 7	$5	USENIX
USENIX	Salt Lake	Summer '84	$10	$25	$5	USENIX

Local User Groups

The USENIX Association will support local user groups in the following ways:

• Assisting the formation of a local user group by doing an initial mailing for the group. This mailing may consist of a list supplied by the group, or may be derived from the USENIX membership list for the geographical area involved. At least one member of the organizing group must be a current member of the USENIX Association. Membership in the group must be open to the public.

• Publishing information on local user groups in *;login:* giving the name, address, phone number, net address, time and location of meetings, etc. Announcements of special events are welcome; send them to the editor at the USENIX office.

Please contact the USENIX office if you need assistance in either of the above matters. Our current list of local groups follows.

In the **Atlanta** area there is a group for people with interest in UNIX or UNIX-like systems, which meets on the first Monday of each month in White Hall, Emory University.

Atlanta UNIX Users Group
P.O. Box 12241
Atlanta, GA 30355-2241

Marc Merlin (404) 442-4772
Mark Landry (404) 365-8108

In the **Boulder**, Colorado area a group meets about every two months at different sites for informal discussions.

Front Range Users Group
USENIX Association Exhibit Office
5398 Manhattan Circle
Boulder, CO 80303

John L. Donnelly (303) 499-2600
{boulder,usenix}!johnd

A UNIX users group has formed in the **Coral Springs**, Florida, area. For information, contact:

S. Shaw McQuinn (305) 344-8686
8557 W. Sample Road
Coral Springs, FL 33065

Dallas / Fort Worth UNIX User's Group
Seny Systems, Inc.
5327 N. Central, #320
Dallas, TX 75205

Jim Hummel (214) 522-2324

In **Fresno,** California, a Central California UNIX User's Group is now being formed to bring together UNIX users, programmers, and administrators. Initially the group will consist of an electronic mailing list to which questions, comments, answers, rumors, and tips will be posted. Communication will be by uucp.

Educational and governmental institutions contact:

Brent Auernheimer (209) 294-4373
Department of Computer Science
California State University
Fresno, CA 93740-0109

CSNET: brent@CSUFresno.edu
uucp: csufres!brent

Commercial institutions, contact:

Gordon Crumal (209) 435-4242
Harry J. Wilson & Company
Insurance Correspondents, Inc.
2350 W. Shaw Ave, Suite 110
Fresno, CA 93711

uucp: csufres!tower!gordon

The **Los Angeles** UNIX Group (LAUG) meets on the third Thursday of each month in Redondo Beach, California. For additional information, please contact:

Drew Bullard (213) 535-1980
{ucbvax,ihnp4}!trwrb!bullard

Marc Ries (213) 535-1980
{decvax,sdcrdcf}!trwrb!ries

In **Minnesota** a group meets on the first Wednesday of each month. For information, contact:

UNIX Users of Minnesota
Shane McCarron (612) 786-1496
ihnp4!meccts!ahby
ahby@mecc.com

In the northern **New England** area is a group that meets monthly at different sites. Contact one of the following for information:

Emily Bryant (603) 646-2999
Kiewit Computation Center
Dartmouth College
Hanover, NH 03755

David Marston (603) 883-3556
Daniel Webster College
University Drive
Nashua, NH 03063

decvax!dartvax!nneuug!contact

In the **New York City** area there is a non-profit organization for users and vendors of products and services for UNIX systems.

Unigroup of New York
G.P.O. Box 1931
New York, NY 10116

Ed Taylor (212) 513-7777
{attunix,philabs}!pencom!taylor

The **New Zealand** group provides an annual Workshop and Exhibition and a regular newsletter to its members.

New Zealand UNIX Systems User Group
P.O. Box 13056
University of Waikato
Hamilton, New Zealand

An informal group has started in the **St. Louis** area:

St. Louis UNIX Users Group
Plus Five Computer Services
765 Westwood, 10A
Clayton, MO 63105

Eric Kiebler (314) 725-9492
ihnp4!plus5!sluug

In the **San Antonio** area the San Antonio UNIX Users (SATUU) meet twice each month with the second Wednesday being a dinner meeting and the third Wednesday being a "roving" meeting at a user site.

San Antonio UNIX Users
7950 Floyd Curl Dr. #102
San Antonio, TX 78229-3955

William T. Blessum, M.D. (512) 692-0977
ihnp4!petro!bles!wtb

In the **Seattle** area there is a group with over 150 members, a monthly newsletter, and a software exchange system. Meetings are held monthly.

Bill Campbell (206) 232-4164
Seattle UNIX Group Membership Information
6641 East Mercer Way
Mercer Island, WA 98040

uw-beaver!tikal!camco!bill

A UNIX/C language users group has been formed in **Tulsa**. For current information on meetings, etc. contact:

Pete Rourke
$USR
7340 East 25th Place
Tulsa, OK 74129

In the **Washington, D.C.,** area a new group is forming. For information contact:

Samuel Samalin (703) 448-1908

USENIX Association
P.O. Box 2299
Berkeley, CA 94710

First Class Mail

Calls for Papers

RT PC Distributed Services: File System

4.3BSD Manuals Available to All Members

2.10BSD Now Available

Work-in-Progress Reports

Change of Address Form

Please fill out and send the following form through the U.S. mail to the Association Office at the address above.

Name: _____ Member #: _____

OLD: _____ _____

_____

_____

_____

NEW: _____ _____

_____

_____

Phone: _____

uucp: {decvax,ucbvax}! _____